Методичка 2.3 - Адресация и смещения

Введение в адресацию

Исходный код программы prog-1.3.1.asm

```
.386
.model flat, stdcall

_data segment
message1 db 'AABBCC',0
message2 db 'DDEEFF',0
_data ends

_text segment
start:
xor eax, eax
mov eax, offset message1
xor ecx, ecx
mov ecx, offset message2
ret
_text ends
end start
```

В данной программе 2 сегмента. Сегмент с данными - _data и сегмент с кодом - _text.

Сегмент с данными содержит 2 строки - 'AABBCC' и 'DDEEFF'. Строки завершается нулевым байтом. Это необходимо для того, чтобы можно было определить, где строка заканчивается.

Перед строкой есть имя строки - message1 и message2, а также тип данных db.

Тип данных DB означает - Define Byte.

Это означает, что указанные данные состоят из одиночных байт. Другими словами, 1 символ - это 1 байт.

Бывают и другие типы: DW - Define Word (2 байта), DD - Define Double (4 байта).

В сегменте кода происходит обнуление регистра EAX с помощью оператора XOR. Далее выполняется оператор MOV, который загружает адрес строки message1 в регистр EAX во время выполнения программы.

Важно понимать, что на этапе трансляции адрес строки не может быть определен транслятором. Транслятор может узнать только смещение для этой строки.

Для получения смещения используется оператор OFFSET, который позволяет получить смещение для указанных данных (например, message1 или message2) относительно начала сегмента данных.

Далее, уже в процессе линковки, осуществляется вычисление адреса строки на основе смещения, которое хранится в объектном файле.

Компиляция.

```
> ml /c /coff prog-1.3.1.asm
> link /SUBSYSTEM:CONSOLE /DYNAMICBASE:NO prog-1.3.1.obj
```

Получаем файл prog-1.3.1.exe.

Запускаем отладчик OllyDbg и открываем программу prog-1.3.1.

Переходим к секции кода (Ctrl + G, 0x00401000).

Команд offset message1 и offset message2 уже нет. Во время линковки были вычислены адреса точного расположения строк в памяти.

```
00401002   B8 00304000      MOV EAX,prog-1_3.00403000            ; ASCII "AABBCC"

00401009   B9 07304000      MOV ECX,prog-1_3.00403007            ; ASCII "DDEEFF"
```

Обратите внимание, что в опкодах адрес 0x00403000 в обратном порядке - 00304000.

Дело в том, что процессоры с архитектурой x86 работают с операндами в обратном порядке. Этот порядок байт называется little-endian (от младшего к старшему).

Получились следующие адреса: 0x00403000 и 0x00403007. Разница между ними составила 7 байт (6 букв и нулевой байт в конце).

Создание циклов

Для того, чтобы выполнить определенный набор инструкций несколько раз, удобно использовать циклы. В языке Ассемблер циклы реализуются с помощью оператора LOOP, а в регистре ECX хранится количество проходов в цикле.

Исходный код программы prog-1.3.2.asm

```
.386
.model flat, stdcall

_text segment
start:
xor eax, eax
mov ecx, 3
label_1:
inc eax
loop label_1
ret
_text ends
end start
```

В программе prog-1.3.2 реализован цикл, в котором 3 раза выполняется инкремент значения в регистре EAX. Инкремент - это увеличение на единицу.

Компиляция:

```
> ml /c /coff prog-1.3.2.asm
> link /SUBSYSTEM:CONSOLE /DYNAMICBASE:NO prog-1.3.2.obj
```

Получаем файл prog-1.3.2.exe.

Запускаем отладчик и открываем программу prog-1.3.2.exe.

Пошагово выполняем программу (F8).

Счетчик (значение в регистре ECX) с каждым проходом уменьшается. В данном случае значение в регистре ECX изменяется от 3 до 0. Стоит отметить, что метка label_1, которая используется в коде, отсутствует. Метки используется только при трансляции программы.

Оператор LOOP выполняет переход по метке только в том, случае, когда регистр ECX не равен нулю.

Безусловные переходы

Безусловный переход - это передача управления по другому адресу или переход по метке независимо от условий, т.е. всегда.

Для безусловного перехода в языке Ассемблер используется оператор JMP.

Условные переходы

Условный переход - это передача управления по другому адресу или метке при соблюдении определенного условия. В случае использования оператора LOOP, условием будет являться - ECX == 0. Если ECX не равен нулю, то переход осуществляется, если равен, то выполняется следующая инструкций после LOOP.

Условных переходов в языке Ассемблер достаточно много. Вот наиболее распространенные операторы условных переходов:

jz / je	- ноль или равно	- ZF = 1
jnz / jne	- не ноль или не равно	- ZF = 0
jle / jng	- меньше или равно / не больше	- ZF = 1

На первый взгляд, кажется, что их очень сложно запомнить, но если разобраться, что означает каждая буква, то всё становится проще.

Оператор JZ - это jump if zero, т.е. прыгать, когда флаг равен нулю.
Оператор JNZ - это jump if not zero, т.е. прыгать, когда флаг не равен нулю.
Оператор JE - это jump if equal, т.е. прыгать, когда равно.
Оператор JLE - это jump if less or equal, т.е. прыгать, когда меньше или равно.

Оператор сравнения

Операторы условных переходов принимают решение на основе флагов, а значит состояние флагов кто-то должен менять. Выполняет проверку условия и по результатам меняет состояние флагов оператор CMP.

Пример использования данного оператора:

```
cmp eax, 0
```

Выполняется сравнение значения в регистре EAX с нулем. Результат сравнения с нулем всегда хранится во флаге ZF (Zero Flag). Если регистр EAX будет равен нулю, то сравнение с нулем будет положительным и ZF будет выставлен в 1. Далее оператор JZ будет ориентироваться на значение в регистре ZF.

Исходный код программы prog-1.3.3.asm

```
.386
.model flat, stdcall

_text segment
start:
xor eax, eax
label_1:
```

inc eax
dec eax
cmp eax, 0
jz label_2
add eax, 4
ret
label_2:
ret
_text ends
end start

В программе prog-1.3.3 реализован условных переход. В начале программы идут инструкции, которые выполняют арифметические операции со значением в регистре EAX, а затем происходит сравнение получившегося значения с нулем. Если значение будет равно нулю, то произойдет переход по метке label_2 и инструкция "add eax, 4" не будет выполнена, а если значение в регистре EAX не будет равно нулю, то инструкция "add eax, 4" будет выполнена.

Компиляция:

> ml /c /coff prog-1.3.3.asm
> link /SUBSYSTEM:CONSOLE /DYNAMICBASE:NO prog-1.3.3.obj

Получаем файл prog-1.3.3.exe.

Запускаем отладчик и открываем программу prog-1.3.3.exe.

Пошагово выполняем программу (F8).

Значение в регистре EAX с каждым шагом меняется. Затем происходит сравнение с помощью оператора CMP. После выполнения этой инструкции значение флага ZF стало равно 1.